SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

METHOD AND APPARATUS FOR SECURE PROCESSING OF SENSITIVE DATA

Background of Invention

[0001] *FIELD OF THE INVENTION*

[0002] The present invention relates to a system and method for the secure processing of sensitive data. More specifically, the invention relates to systems where parties, which do not necessarily trust each other, need to control the processing and the use of their data.

[0003] *BACKGROUND OF THE INVENTION*

[0004] Many IT applications are comprised of the processing of sensitive data, i.e. data that should be kept private or confidential by the IT system. A typical example for such an application is the screening of individuals against watch lists prior to boarding an airplane, or prior to employment. Such a screening can be a simple textbased matching of names and other personal data, especially, biometrics like fingerprint minutiae or features of the human face, (see Anil Jain, Lin Hong and Sharath Pankanti, "Biometric identification", Communications of the ACM, 43 (2), pages 90-98, ACM Press, 2000), can be used to confirm the identity of matched individuals or to identify individuals in a set of candidates.

[0005] Cryptographic techniques exist to encrypt and protect data while it is sent over computer networks or stored on storage media, (see for example the book Bruce Schneier, "Applied Cryptography", John Wiley & Sons, 1996). However, for the real processing steps, data has to be available in the clear to allow a computer system to interpret, and modify them, and to derive new data during the processing. During this

processing, data is vulnerable to attackers, who intrude into the processing environment to spy out or modify data. IT systems have no means to protect programs and data against an attacker who found a way into their processing environment.

[0006] A way to protect an IT system against intruders is to confine the processing of sensitive data to a secure computation environment, (see Schneck, et al., "System for controlling access and distribution of digital property", U.S. Pat. No. 6,314,409, issued Nov. 6, 2001, hereby incorporated herein by reference, and Sean W. Smith and Dave Safford, "Practical Private Information Retrieval with Secure Coprocessors" hereby incorporated herein by reference, IBM Research Report, RC 21806, July 2000). A secure computation environment is a general-purpose computing device like the IBM 4758 cryptographic coprocessor, which has a FIPS 140-1 Level 4 validation, (see http://www.ibm.com/security/cryptocards). Computations in a secure computation environment cannot be observed from the outside. Further, it is impossible to maliciously change the processing or the processed data from outside the secure computation environment. Attempts to tamper with a secure computation environment, the programs running inside, or the data being processed inside are detected by the environment, which then destroys sensitive data stored inside the environment or makes it permanently inaccessible.

[0007] While a secure computation environment can give protection against attacks from the outside, it cannot guarantee by itself that the security and privacy demands of the parties are met by the system. The parties have to trust the provider of the application that their data is handled in the way they expect. There is no technical guarantee for this, because the parties cannot control the service.

[0008] Often parties are not willing to participate in a data processing task if they cannot control the processing and the use of their data, i.e. if they have to rely totally on trust.

[0009] Existing systems do not allow the involved parties to control the service, or they simply reduce the number of parties to one to circumvent the problem. For example, some airlines have databases of their own unruly passengers, and they can check passengers against this data. Here, all roles, i.e. the service provider, the information

provider, the screener and the notified parties are all represented by the airline, which naturally trusts itself. Other examples of databases are law enforcement agencies, (see James X. Dempsey, "Overview of current criminal justice information systems", in Proceedings of the 10th Conference on Computers, Freedom and Privacy, Toronto, pages101-106, ACM Press, 2000), or credit history checks. These applications are under the complete control of the information provider, who is also the service provider. The screener party has no way to control how the service provider further uses the information that the screener provides. A similar example is a face recognition application at an airport, where passengers are checked against the picture in their own ID document. This application is under complete control of the screener, who is also the service provider.

[0010] Existing surveillance systems, as for example the facial recognition system used at the 2001 SuperBowl, or in several airports, (see Scholz and Johnson, "Interacting with identification Technology: Can it make us more secure?", in Conference on Human Factors in Computer Systems, Minneapolis, pages 564-656, ACM Press, 2002), are mostly closed circuit systems. They return a direct answer to queries that compare biometric data against a local database, or against information stored on a smart card, which is carried by the individual. Thus, there is a need for a more general service that can be used by different interacting parties.

[0011] Other systems, such as are described in U.S. Pat. No. 6,148,290, issued Nov. 14, 2000 to Asit Dan and Francis Nicholas Parr, hereby incorporated herein by reference in its entirety, have been used to define possible interaction patterns with computer programs, where the interface specifications defines sequences of requests that are admissible to a server. In these systems, the service provider dictates a service contract. The contract defines assertions for the interface of the service, e.g. that a request A must be preceded by another request B to be valid. It is not possible for the parties to use these kinds of contracts to control the processing or the use of their data.

[0012] Most existing systems do not execute in secure computation environments. Here, an attacker who gains access to the processing may observe and influence the processing at will. Thus, even if the parties agree on processing steps and handling of

data, there is no technical way to guarantee that the processing fulfills their security requirements in such an insecure environment.

[0013] Existing systems built on web services (Vadim Draluk, "Discovering Web Services: An Overview", in Proceedings of the Twenty-seventh International Conference on Very Large Data Bases: Roma, Italy, page 637 Morgan Kaufmann, 2001) allow their clients to control which service provider they choose, but they allow no control over the service"s functionality itself.

[0014] Existing systems that use explicit contracts, like e.g. service level agreements or quality of service agreements as described in (Dinesh Verma, Mandis Beigi and Raymond Jennings, "Policy Based SLA Management in Enterprise Networks", in Lecture Notes in Computer Science, Vol. 1995, pages 137 152, Springer-Verlag, 2001), only deal with nonfunctional features of services like resource consumption or turnaround times. These approaches do not allow parties to control the function of the service.

Summary of Invention

[0015] The present invention allows multiple parties, which do not necessarily trust each other, to participate in the processing of sensitive data. The relationship between the parties is secured by machine-interpretable service specifications (also called contracts) that can be automatically enforced by the processing system. The machine-interpretable service specifications define how a request is to be processed and how the information provided by the single parties is used and disclosed in this process. The processing is preferably executed in a secure computation environment, which ensures that nobody can observe the processing or tamper with it.

[0016] In one embodiment of the invention, service specifications may be agreed upon by all parties before they are implemented into the system. In this way, the parties who use the service for the processing of sensitive data have full control over the processing and the use of their data.

[0017] In another embodiment of the invention a requestor triggers the data processing by sending a service request to a server. The processing itself is neither observable nor can it be tampered with from outside. At the end of the processing, the service sends notifications to some or all the parties, which were directly or indirectly involved

in the processing of the request, which may also include the requestor.

[0018] In still another embodiment of the present invention the processing of service requests is defined by service specifications, which specify the actions to be taken during the processing of a request, and how information flows between these actions. The parties who cooperate in the processing of a service request must mutually agree on a service specification before the system will execute it.

[0019] In yet another embodiment of the present invention the parties that cooperate in the processing of a request are owners of data items that are computed during the processing of a request. The owner of a data item can precisely control, to which processing step or to which other party he wants to disclose a data item.

[0020] Other features and advantages of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, which give formal specifications as well as examples of the principles of the invention.

Brief Description of Drawings

[0021] Fig. 1 illustrates a general workflow between participating parties according to an embodiment of the present invention.

[0022] Fig. 2 shows a block diagram of a global structure of the system for secure data processing defined and controlled by service specifications according to an embodiment of the present invention.

[0023] Fig. 3A shows a structure of service specifications as a class diagram in the Unified Modeling Language (UML) according to an embodiment of the present invention.

[0024] Fig. 3B shows a UML specification of ItemSets and ItemSpecSets according to an embodiment of the present invention.

[0025] Fig. 4A shows a flow chart diagram for the processing of the Request Handler, the conditional execution of the processing steps and the conditional generation of the notifications according to an embodiment of the present invention.

[0026] Fig. 4B shows a state diagram of the negotiation of a machine interpretable service specification.

[0027] Fig. 4C shows a state diagram for uploading, activation, and revocation of a machine-interpretable service specification.

[0028] Fig. 5 shows the integration with network and secured database in a service cell according to an embodiment of the present invention.

[0029] Fig. 6 shows an example contract for an application, where individuals are securely checked against watch lists according to an embodiment of the present invention.

[0030] Fig. 7 illustrates an example of a possible distribution according to an embodiment of the present invention.

[0031] Fig. 8 illustrates different security spheres according to an embodiment of the present invention.

[0032] Fig. 9 shows a screening scenario according to an embodiment of the present invention.

[0033] Fig. 10 shows a general structure of an embodiment according to the present invention.

[0034] Fig. 11 shows a functional modular diagram according to an embodiment of the present invention.

[0035] Figs. 12A – 12C show different possible designs according to other embodiments of the present invention.

Detailed Description

[0036] The present invention defines a method for the secure processing of sensitive data as follows: a) The parties, which cooperate in the processing of sensitive data, first develop a relationship in which they agree on the functional steps that comprise the processing and on the data items that should be used in the single processing steps.

[0037] b) The parties agree on a formal specification of the processing and the use of

data, which is manifested in a machine-interpretable service specification (contract). Among other information, the service specification defines the single processing steps of the processing, which data items should be used as input for a single processing step and what output the processing step contributes to the processing state.

[0038] c) The service is an interpreter for service specifications. It guarantees that the system executes exactly the functionality defined by the service specification and uses the data items only in the intended way. Especially, this means that the processing and the data items used cannot be observed or tampered with from the outside.

[0039] For a screening scenario, the involved parties might be the service provider, who operates the screening service, the screener, who wants to check an individual, several information providers, which provide the watch list data, and parties like law enforcement, airport security or human resources, which should be informed in case the screened individual is found on a watch list.

[0040] The information provider wants to keep his watch list data confidential, i.e. he allows its use for screening, but the information on the lists must not leak outside the system. The screener wants to guarantee basic privacy for the screened individuals as long as they do not show up on the watch lists. In particular, he does not want to enable the information provider to trace arbitrary individuals based on screening requests. Screener and information provider only want to notify law enforcement in case of a match and do not want to leak information in other cases. None of the parties wants that information leaks to the service provider or to intruders.

[0041] Accordingly, an apparatus for secure processing of sensitive data includes a network server, which receives a request for processing from a requestor over a secure network channel, a coprocessor based secure computation environment for processing sensitive data in a way that can not be externally observed or tampered with, and the contract engine that controls the processing of the request based on service specifications on which the parties agreed. The contract engine includes a request handler, which selects the service specification that governs the request, and checks whether the party is allowed to send the request to the server, and whether the request conforms to the service specification. A step processing unit, which performs a sequence of processing steps as defined in the service specification, thereby using

and extending the processing state, which initially contains the information sent in the request. A notification generator, which selects parts of the processing state as defined in the service specification, and generates notifications, which it sends over secure network channels to the parties defined in the service specification.

[0042] Fig. 1 illustrates a general workflow between participating parties according to an embodiment of the present invention. In general, all processing is expressed by the following scheme: A distinguished party, the requestor 101 sends a service request 110 to the secure request processing 102. The request processing 102 determines the service specification, which governs the processing of this request and executes a sequence of processing steps based on that service specification. The request processing 102 can also receive requests or data from another party 103 which might be based upon a different service specification. The application of a single processing step uses a subset of the processing state as input. The output of the processing step is used to extend the processing state. Input and output specifications are defined in the service specification. The initial state of the processing consists of the information from the request that triggered the processing. When all applicable processing steps have been executed, the request processing 102 generates notifications 111 for parties 101, 103 and 104 that are involved in the processing of the request. Whether a notification is sent and the contents of the messages for the parties are specified in the service specification.

[0043] Each request can have a different requestor. For example after request 110 is processed in Fig. 1, party 103 can take on the role of the requestor and send its own service request, 112. Different kinds of service requests invoke different service specifications, which in turn provide different service functionality to the clients.

[0044] In a typical application, an information provider, 103, would from time to time send service requests, 112, to update the information that he provides to the service. The service specification that controls this kind of request specifies how the data in the request from the information provider is used to change the information inside the service. Other service requests will thereafter be processed using the updated information.

[0045] The system allows different parties to interact at the same time based on different

service specifications. For example, in a screening application, there can be a multitude of different screener parties, different information providers, and different law enforcement agencies. Each request sent to the system invokes processing based on a specified service specification. The service specification determines which parties interoperate in the processing of this request. For convenience, we also define composite contracts, see further below.

[0046] Because the processing is controlled by service specifications, and the participating parties control these service specifications, they have complete control over the processing and the way their data is used. Executing the service specification in a secure computation environment guarantees that information does not leak and that no one can tamper with the processing of the data.

[0047] Fig. 2 shows a block diagram of a global structure of the system for secure data processing defined and controlled by service specifications according to an embodiment of the present invention. Generally, the processing starts when the server, 201, receives a service request, 200, over a network connection, 204. The request, 200, is processed by the contract engine, 203, which consists of the request handler 205, the step processing 208 and the notification generator 209. All three components work with the processing state 207 and are controlled by a service specification 206.

[0048] The request handler 205 first checks whether the incoming request is valid and admissible for processing. It does this by checking the content of the request and the party who sent the request against information defined in the service specification, 206, which defines and controls the processing of this particular service request.

[0049] In the system which is described by the present invention, the processing of service requests is controlled by service specifications, 206. A service specification (See Fig.6) is a formal specification, which is mutually agreed upon by all parties, who cooperate in the processing of a service request. A service specification precisely defines and controls how a server processes a request and how information is disclosed and aggregated between single processing steps during this processing.

[0050] A service specification specifies:

[0051] a) the parties that cooperate in processing a request;

[0052] b) the data items that are provided in the request message;

[0053] c) the single actions that have to be executed during the processing of a request;

[0054] d) the data items that are generated as the result of executing an action, and the ownership of these data items;

[0055] e) the workflow, which controls the application of single actions; and

[0056] f) the controlled data flow between single actions, which allows the owner of a data item to precisely control, to which processing steps he wants to disclose his data, and to which he does not.

[0057] A formal data model and more details about service specifications are given below.

[0058] If the check of the incoming service request 200 by the request handler 205 fails, the request is turned down. Otherwise, if everything is acceptable, the processing continues with the processing of the single processing step in the step processing 208.

[0059] The step processing 208 uses two data structures. The service specification 206 on which the processing is based, and a processing state 207, which accumulates data items that are produced by the single processing steps during the course of the request processing. The processing state 207 is an ordered set of name/value pairs, called an ItemSet, see Figs. 3A and 3B.

[0060] Initially, the processing state 207 of the step processing 208 consists only of the data items, which had been provided by the incoming service request 200. The step processing 208 uses the service specification 206 to determine a sequence of processing steps that should be executed to process the service request. The step processing 208 also determines from the service specification which part of the processing state should be fed into the single processing steps that comprise the processing of the service request. The results of a processing step, also defined in the service specification, are then used to extend the processing state and are therefore

available for the next processing step. While the step processing only reads the service specification, it does read and modify the processing state 207. The step processing 208 ends, when all processing steps, which the service specification specifies, are processed in the specified order.

[0061] After all processing steps have been processed by the step processing 208, control is passed to the notification generator 209. The notification generator mainly generates the notifications 210 that are sent out to the cooperating parties. Notifications are generated from the processing state. In fact, the notification generator 209 is similar to the step processing 208. The main difference is that the processing steps executed by the notification generator 209 are special because they have predefined semantics.

[0062] In its electronic form, a service specification 206 can be interpreted and executed by a computer program. If executed in a secure computation environment 202, the electronic execution of a service specification guarantees that all steps are processed in the specified order and that information exactly flows, where the service specification states it should.

[0063] A secure computation environment 202 is a computer that is sealed, so its internal processing is not observable from the outside, and no one on the outside can modify the computation or the data inside the device without proper authorization. Such a device detects attempts to open, intrude or probe it. If any of these tamper attempts are detected, the secure computation environment destroys sensitive data stored inside and ceases operation completely. An example of such a secure computation environment is the programmable IBM 4758 cryptographic coprocessor.

[0064] The security features of secure computation environments 202 can be extended to communications over computer networks. This can be achieved by using standard cryptographic techniques on the network communications between two secure computation environments. Communication sessions secured in this way, also known as channels, 204, prevent an attacker from reading data communicated over the channel. They also make it possible to detect when an attacker tries to modify messages, insert his own messages, or impersonate one of the secure computation environments. Of course, the network is not trusted, and all cryptographic techniques

have to be implemented inside the secure computation environments. For more details on channels see the description of Fig. 5.

[0065] It is worth noting, that if the system described in the present invention would not be implemented within a secure computation environment, it would be very vulnerable to a variety of attacks. In particular, in an insecure environment, electronic execution of a service specification can not guarantee, that information flows only to the specified processing steps and is never disclosed or modified, because the request 200 and the processing state 207 would be accessible to an attacker.

[0066] Fig. 3A shows a structure for service specifications as a class diagram in the Unified Modeling Language (UML) according to an embodiment of the present invention. We call this implementation of a service specification a contract and use the terms contract and service specification as synonyms. Each contract 307 has a contract identifier (identifier), which uniquely distinguishes this contract from all other contracts in the system. A contract 307 specifies one or more parties (parties), who will cooperate in fulfilling a service request. The Party class 301, which describes a single party, includes a unique identification of the party in the system (id), and a network access point, to which the service can send messages in response of the processing of a service request. A service access point might for example be defined by an Internet address and a port number, or by an uniform resource locator (URL). It also includes the name of the party. In one embodiment, this minimal definition of a party is most likely augmented by further attributes, like a street address, which are not required for the secure processing of requests, but might be necessary for maintaining additional relationships among the parties.

[0067] The contract 307 further specifies a single action (trigger), which is the arrival of a service request that should be processed. An object of the RequestDescription class 304 describes the incoming request, which triggers the processing as defined and controlled by this contract. In particular, the RequestDescription defines what data fields have to be present in the request.

[0068] The contract 307 still further specifies zero or more processing steps specification (steps), which are executed during the processing of the request. The specifications of the processing steps are given as an ordered list modeled by ProcessingStepSpec

objects, 305. Furthermore, it specifies zero or more outgoing messages (notifications), with which the system responds to the cooperating parties after processing of a request. The notifications are given as an ordered list of NotificationSpec objects 306. An Action 302, is the common abstraction of RequestDescription 304, ProcessStepSpec 305, and NotificationSpec 306. Each action 302 has a name (name), and a specification (outputSpec) of an ordered set of data items (ItemSpecSet), which it provides after completion. It is owned by a single party (owner).

[0069] There are multiple actions that can be performed. A first type of action is the arrival of a new service request. The owner of the RequestDescription 304 defines the requestor, i.e. the party, which can send this kind of requests. The outputSpec of a RequestDescription 304 action describes the data items that are present in the request. Each request has at least one mandatory item called identifier, which is used to uniquely identify the contract that defines and controls the processing of this request.

[0070] A second type of action is a Function 303. A function 303 takes an ordered set of data items as input (inputSpec) and returns an ordered set of data items as output (outputSpec). The execution of a function is controlled by a Boolean expression (condition). The Boolean expression is an expression that is constructed from the special constants 'true' and 'false', from the Boolean operators 'not', 'and', and 'or', the relational operators <, <=, =, /=, >, >=, from arbitrary constants, and from names of items in the processing state, see Fig. 2. The function is only executed if its condition yields true.

[0071] A Processing Step Specification 305 is a function, which is defined by one of the cooperating parties (its owner), and which can execute arbitrary functionality. The functionality to be executed is defined by referencing the code that implements the processing step.

[0072] A Notification Specification 306 is like a Processing Step Specification 305, but the functionality provided by a notification is predefined and fixed. It cannot be changed by the cooperating parties. Likewise, the number of items in the Notification"s output specification is fixed to one. However, the name of this return code can be freely

defined.

[0073] Fig. 3B shows a UML specification of ItemSets and ItemSpecSets according to an embodiment of the present invention. ItemSpecSets 311 are used to define inputSpecs and outputSpecs. An ItemSpecSet 311 defines an ordered set of data item specifications called ItemSpec 312. A data item specification identifies a data item by its name (name). A data item defined by the Item class 314 is a name/value pair. An ItemSet 313 is an ordered set of data items.

[0074] Fig. 4A shows a flow chart diagram for the processing of the Request Handler, the Step Processing, and the Notification Generator according to an embodiment of the present invention. An incoming request can be an item set as defined in Figs. 3A and 3B. When the server receives a new service request, in step 400, the request handler first extracts the mandatory item identifier from the item set that constitutes the request. The value of the identifier item allows the request handler in step 401 to retrieve the contract (service specification), which controls the processing of the incoming service request. Then in step 402, the request handler checks whether it accepts the request from this sender. For this check, the request handler selects the party that owns the trigger of the contract and compares it to the party on the other end of the channel over which the request was received. If both values denote the same party, the request is considered for further processing; otherwise the processing terminates in step 403 with the error handler.

[0075] The next check, in step 404, determines whether the request is complete, i.e. whether it provides all the items that are required by the contract. This check is performed by asserting that all names that are defined in the output specification of the trigger occur exactly once in the request"s item set and that they all have defined values. The order, in which the items occur, does not matter. If the request has additional items that are not described in the trigger output specification, this is not considered an error. Since none of the following processing steps can access those items, they are simply ignored by the request processing and never compromised. In case of an error, the request processing terminates in step 403.

[0076] If both checks are passed, the request handler initializes the processing state S for the step processing in step 405. The initial processing state contains exactly the

items, for which the names are specified by the trigger output specification.

[0077] The step processing starts with step 410. While there are more steps in the ordered list of processing steps in the contract, the step processing selects the next processing step PS in order in step 411.

[0078] Step 412 evaluates PS"s condition over the processing state S. Processing step conditions can access all items of the processing state. Since the expression being evaluated is obvious from the contract to all parties, they obviously already agreed on the particular use of items from the processing state for the evaluation of a given condition. If the condition yields false, the processing step PS is ignored and the processing continues with the next processing step specification in the list of processing steps in 410. If the step"s condition evaluates to true, step 413 builds a new item set s. The processing inserts items from the processing state S into the new item set s, if it finds their name in the input specification of processing step PS.

[0079] Step 414 determines the processing step implementation from PS"s step specification and then calls it with the new item set s as its only parameter. The result of this call is the item set r. Step 415 takes all items from r, for which the names occur in the output specification of processing step PS and inserts them into the processing state S. Step 415 never overrides existing values in S with new values from r. If an item with the same name already exists, it must also have the same value; otherwise the contract is not legal. However, several processing steps may specify the same item specifications in their output specifications, if the conditions of the processing steps or their implementations guarantee that two activated steps never produce different values for the same item name.

[0080] Once all of the processing steps are processed, the step processing unit passes control to the notification generator. This is shown in step 420. The generation of notifications follows the same basic structure as the execution of the processing steps.

[0081] While there are unprocessed notification specifications, they are processed in order. Fetching notification specifications and evaluating their condition in steps 421 and 422 is identical to the processing in steps 411 and 412 of the conditional

execution of processing steps, and step 423 is similar to 413. After generating the new item set s in step 423 from the processing state S and the input specification of the actual notification, N, step 424 fetches the network access point from the party, P, who is defined as the owner of N in the notification specification. The set s is then sent out as a notification to Party P.

[0082] The output specification for each notification specification is a set with a single element. The value of this element is Boolean and states, whether the notification was successfully delivered to the party P. The name of this item can be freely defined. The item is added to the processing state S and is therefore accessible for other notifications that are not yet processed. This makes it in particular possible to send out a notification depending on whether another notification failed or succeeded, or to include return codes of former notifications in other messages. Once all notification specifications are processed, the whole request processing terminates successfully in step 426.

[0083] Even though the sequential processing of processing steps and notifications has the benefit of being simple, it might not always allow for very efficient processing. One embodiment chooses to determine the dependencies between actions, and executes independent actions in parallel once all their items given in their input specifications are available.

[0084] Fig. 4B shows a state diagram of the negotiation of a machine interpretable service specification. Negotiation of a machine-interpretable service specification is a process between all parties that cooperate in the processing of the according service request. The process starts in an initial state 450. A proposal for a service specification by one of the parties leads to state 451. In state 451 some or all parties interact in the definition of the contract by proposing additional modifications to the actual specification. Agreement to the specification by one party ends the definition phase and makes a transition from state 451 to state 452. State 452 awaits the agreement to the defined proposal by all parties. If in state 452 a party makes a new proposal to modify the already defined contract, the process goes back to state 451. After all parties agreed in state 452, negotiation of the specification ends in state 453.

[0085] Before parties can cooperate in the processing of a service request, they first have

to agree on a service specification that defines and controls the basic, initially defined and agreed upon processing of this kind of request. Only after all parties have agreed, is the processing, which is based on this service specification, activated in the server. The set of activated service specifications can be dynamically extended by the parties. By installing and activating new service specifications the service learns how to process new kinds of requests.

[0086] Fig. 4C shows a state diagram for uploading, activation, and revocation of a machine-interpretable service specification. Starting from an initial state 461, the process makes a transition to state 462 when a service specification is uploaded into the system. State 462 awaits activation of this specification by all parties that negotiated the specification. Once all parties activated the uploaded specification, the process goes to state 463. In state 463, the uploaded specification is enforced by the system. It starts to accept and process service requests according to this specification. More than one specification can be loaded into the system at one point of time. If multiple specifications are active, the service request must identify the specification that should be used for the processing of that request. If at least one party revokes the specification, the process enters state 464. In state 464, the specification is cancelled and no further service requests for this service specification will be accepted by the system.

[0087] A service specification can be considered a formal specification, which can be represented as a data structure, as an electronic document. In one embodiment, the activation of a service specification by mutual agreement is implemented in a way where all parties upload electronically signed copies of the service specification to the server. After the server receives such signed copies from all parties, verifies the electronic signatures, and validates, that all copies indeed specify the same service specification, the processing is automatically activated.

[0088] Similarly, electronically submitted service specifications can also be cancelled at any time or when they expire. If the parties agree as such, a single party may revoke a service specification or if it expires, the system ceases to honor requests governed by this service specification.

[0089] Fig. 5 shows the integration with network and secured database according to an

embodiment of the present invention. The aggregation of a secure computation environment 506 with one or more secure coprocessors 507, a network handler 503, and mass storage handler 505, and one or more secured local databases 504 on the same host computer is also called a service cell or short: a cell. Secure computation environments, like e.g. the IBM 4758 programmable, cryptographic coprocessor, might have no way to directly access a database or a network. Fig. 5 shows how the application code inside the secure computation environment, 506, can securely access a database, 504, and a network, 502. All messages that are sent to the network, 502, are protected by cryptographic techniques, as are all data items stored in the database, 504, like e.g. service specifications and watch lists.

[0090] The host computer, 501, hosts the secure computation environment, 506, the network handler, 503, the mass storage handler, 505, and a database, 504. The network handler, 503, uses calls to the operating system of the host computer, 501, to access the network, 502, and to process network requests that come from inside the secure computation environment. The mass storage handler, 505, uses a standard database interface like SQL to access the database, 504, and to process database requests that come from inside the secure computation environment.

[0091] The network handler, 503, sends and receives messages to and from the network, 502. Messages are protected using cryptographic techniques, i.e. encryption, authentication, message digests, and sequence numbers. This ensures that an observer can neither interpret nor alter the contents of the messages without being detected. The network handler, 503, receives an incoming, encrypted message from the network, 502, and gives it to the channel handler, 508, inside the secure computation environment, 506. The channel handler, 508, checks the cryptographic checksums, removes sequence numbers and decrypts the data. The request processing, 509, inside the secure computation environment, 506, can now access the data in the clear.

[0092] When the request processing, 509, wants to send out a message over the network, it sends it to the channel handler, 508, which encrypts it and adds sequence numbers and message digests. The channel handler, 508, then sends the encrypted message to the network handler, 503, which is on the host, 501, but outside the secure

computation environment, 506, which in turn sends it to the network.

[0093] If the request processing, 509, wants to access the database, 504, it sends a clear-text request to the channel handler, 508, which encrypts the data in the request and sends it to the mass storage handler, 505, which is on the host, 501, but outside the secure computation environment, 506. The mass storage handler 505, can directly access the database, 504, and returns results from the database to the channel handler, 508, inside the card, where the answer is decrypted and returned to the contract engine.

[0094] As an optimization, the request processing, 509, can store parts of the data from the database, 504, in a cache, 510, inside the secure computation environment, 506. This can avoid costly round-trips through the channel handler, 508, the mass storage handler, 505, the database, 504, and back through the mass storage handler, 505, and the channel handler, 508.

[0095] Fig. 6 shows an example contract 601 for an application, where individuals are securely checked against watch lists according to an embodiment of the present invention. Fig. 6 gives an example for a contract, 601, that can be used to check individuals against watch lists of criminals. The information given in the table corresponds to the variables defined in the UML class diagrams in Figs. 3A and 3B. In this example, the contract identifier, 602, is "4711B5" and the involved parties, 603, which are named A, B, and C, are an airport, 603 A, which wants to screen applicants, a government agency, 603 B, which provides watch lists to be checked against, and a law enforcement office, 603 C, which should be notified, when a match is found.

[0096] The rows in the table list the single data items produced by the actions that are defined in the contract. For each block of data items, the first column describes the action that produces the data items, the owner of this action and the condition, under which the action is performed. The columns describe for each processing step and each notification, which data items are passed as inputs.

[0097] The rows 604 define the service request. Field 605 states that only party A can send the request. The fields, 606, in the second column of the rows 604 state which data items have to be present in the request. Rows 607 define the processing steps of

this service specification. In this example, there are three processing steps: "1:n Matching", rows 608, "Biometric Match", row 609, and "Decoy Generation", row 610. The fields 611 in the second column of rows 608 define the data items in the output specification of the processing step "1:n Matching". The fields in the second column of rows 609 and 610 define output specifications with only a single data item for their processing steps. The rows 612 define the four possible notifications of this contract. The fields in the first column of rows 612 define the names, owners and conditions of the notifications, i.e. how the notification is called, to which party it is sent, and under which condition it is sent. The fields in the second column of rows 612 define a single return value in the output specification of the notification.

[0098] The columns 608 define the input specifications for processing steps and notifications. There is a column for each processing step and one for each notification. A plus sign (+) in a field indicates that the data item identified by the row in which the plus sign occurs is part of the input specification of the processing step or notification to which the column belongs. A slash (/) in row a, column b indicates that the data item of row a does not belong to the input specification of the processing step or notification of column b. An empty entry ( ) indicates that it makes no sense to decide whether the data item should be included in this input specification or not, because the data item is not yet computed.

[0099] The example contract 601 is interpreted as follows: The processing is triggered by a request, 604, sent by party A, which contains the contract id, a first name, last name, date of birth, finger print data, a location and a transaction number.

[0100] The first processing step is a 1:n Matching against a watch list named XYZ, 608. This processing step is owned by party B and is executed unconditionally (condition = True). The column named "1:n Matching"defines with +-signs the inputs for this processing step: first name, last name, and date of birth. Outputs are defined in the rows 608: a value stating whether the match was successful, first name, last name, date of birth, a suspect class, fingerprint data from file, information about previous convictions and reaction advice.

[0101] The next processing step 609 employs Biometric Matching. It is owned by party C and is only executed, if the 1:n matching was successful. Inputs are the actual

fingerprint data and the fingerprint data from the 1:n matching step. The output is a value that states the correlation between those two fingerprints.

[0102] Next, the Decoy Generation processing step 610 generates a decoy message, if the 1:n matching was not successful.

[0103] The contract 601 defines in rows 612 three alternative notification to party C, one notification to party A and no notification to party B. All notifications have a single output value, which states whether the receiver acknowledged the notification, or not. Notification1 is sent to C, if the 1:n matching succeeded, and the suspect class of the individual is < 5. It contains first name, last name, date of birth, the location and the output of the biometric matching. Notification2 is sent to C, if the 1:n matching succeeded, and the suspect class is > 4. It contains the fields as Notification2 and additionally a reaction advice. Notification3 is sent in case the 1:n matching failed. It only contains the decoy message, but no data about the checked individual. Notification4 goes to party A. It only contains the transaction number that came in with the actual request and a value that states, whether the service was successful in notifying party C.

[0104] From Fig. 6 it is clear, that the information about previous convictions is not disclosed to any party under this contract. It is also clear, that in case of a failed 1:n match, no information about the individual is sent. (The decoy message holds no real data and is only sent to conceal the fact of a match or no match from an attacker who can observe the communication between the service and party C).

[0105] If a requestor has multiple service specifications, which he wants to execute together, these contracts can be grouped and executed on behalf of a single request. A contract that bundles several other contracts is called a composite contract. The structure of a composite contract is very similar to that of a basic contract. The difference is that a composite contract does not define processing steps and notification, but enumerates the identifiers of the contracts that should be combined. Like a basic contract, the composite contract, too, has a trigger. The output specification of this trigger defines the union of the output specifications of the basic contracts that should be combined in this composite contract. The contract engine processes a composite contract by invoking the subcontracts on the received request.

[0106] In a screening application, for example, this would allow a screener to have a contract for screening against watch lists from agency A and another contract for screening against watch lists from agency B. The combined contract would invoke both searches on behalf of a single request to the system. Other composite contracts may allow checking against yet other combinations.

[0107] Biometric information scanned at an entry point can be sent within a request to one or more servers, which process the data and can check it against multiple databases. The workflow is completely up to the participating parties, including sending an alarm to a law enforcement agency instead of returning data about the screened individual back to the screener.

[0108] Formal proof of the correctness of an implementation of the proposed processing requires verification of the contract engine and the code for the single processing steps. Because all systems built in this way use the same contract engine, verification of the contract engine is a one-time task. Code for single processing steps has to be verified when it is introduced into the system. Code for single processing steps can be reused by different contracts, which minimizes the overall verification effort.

[0109] It is possible to further increase the dynamic features of the described system by making the code of the individual processing steps part of the service specification, in contrast to only providing their names. For techniques that allow the secure downloading of code to the service see for example (Smith, et al., "Securely downloading and executing code from mutually suspicious authorities", U.S. Pat. No. 6,167,521, issued Dec. 26, 2000) hereby incorporated herein by reference in its entirety. Providing the code for the processing steps allows the parties to dynamically extend the functionality of the service and the verification process can be done locally.

[0110] Fig. 7 illustrates an example of a possible distribution according to an embodiment of the present invention. The invention service consists of several, interconnected service cells. A cell may contain one or more secure computation environments, a network handler, a mass storage handler and one or more secured databases, see also Fig. 5. Multiple cells can be installed in one physical location, but there will be more than one physical location as shown in Fig. 7. The cells 801-806 may reside in different locations as shown in Fig. 7. A client can connect to a subset of

the available cells.

[0111] This arrangement allows for sufficient amount of resources, and provides a high degree of fault tolerance of the combined service to single malfunctions. Furthermore, it allows for the ability to easily divide the service and the data according to geography and/or national (legal) concerns.

[0112] Even though the invention can be implemented in smaller scenarios, the architecture has been prepared for global service.

[0113] Fig. 8 illustrates different security spheres according to an embodiment of the present invention. Crypto coprocessors 907, 910 are located in Clients A and B as well as in the service (908, 909, 911). The coprocessors 907 - 911 are physically located with secure storage and communication segments 901 - 906, all of which are located within an outer cell perimeter 920 - 924.

[0114] Therefore, all sensitive operations are confined within the secure cryptographic coprocessors 907 - 911. The database content is encrypted at all times, and integrity is secured by message digests. The Intercell communication is encrypted, authenticated and secured against modifications and replays. Finally, the outer cell perimeter 920 - 924 is secured with firewalls and system level access control.

[0115] Fig. 9 shows a screening scenario according to an embodiment of the present invention. The first one starts with a screening request 1110 from the screener 1101 to a server cell 1103. The cell 1103 receives the request through its network handler 1106 and passes it on into a coprocessor 1108 for processing. If the watch list that should be used for screening is available in this cell, the coprocessor 1108 queries the database 1109 for information about the individual. Then, it uses the network handler to notify law enforcement 1107. This notification is sent in any case. If there is no match, the notification contains a decoy, which is not displayed as an alarm at law enforcement. After the law enforcement 1107 acknowledgement is received, the screener 1101 is notified that his request was processed.

[0116] Another scenario is also likely. Here the main difference is that the watch list is not locally available on the screener"s client cell 1103. In this case, the cell forwards the request to a cell 1105 that has the required table. This cell 1105 also notifies law

enforcement 1107 and returns a notification for the forwarded request, which is returned to the screener 1101.

[0117] Fig. 10 shows a general structure of an embodiment according to the present invention. While it is likely most of the processing is performed inside the coprocessor as shown in Fig. 10 , parts of this invention can be implemented on the host 1204, because the PCI bus 1210 and the serial port are the only ways for the card to communicate with its environment.

[0118] The host application has to provide network access 1224 and mass storage 1226. It also must implement the host part of the card/host communication 1220. An insecure host 1204 includes the host/card interface 1220, the network handler 1224, a mass storage handler 1226, and the database 1228. However, the secure computation environment 1202 includes the coprocessor 1206 and other coprocessors 1208 for handling the main part of the secure processing.

[0119] Fig. 11 shows a functional modular diagram according to an embodiment of the present invention. Fig. 11 shows the basic component architecture, which is comprised of different interfaces and several Cells 1510, 1510', etc. Requests to the system are generated by user interfaces. There are different interfaces for different clients. The service provider has a configuration interface, SP Config, 1570, and a Billing interface 1571 to control and administrate the billing of the service. Screening parties also have configuration interfaces, SC Config 1572, and a graphical user interface to send screening requests, SC GUI. It is likely that the screening party already possesses an IT system, SC Backend, which can automatically generate requests. In this case, a integration component, SC BE Integr, is necessary to hook up that system to the service. The information provider parties also need various interfaces 1573, an integration component, IP BE Integr, to hook up their existing system, IP Backend, to the service. For service configurations, there is a dedicated interface, IP Config. Law enforcement also have configuration interfaces, LE Config 1574, and a graphical user interface, LE Notif GUI, on which alarm messages are displayed. Optionally, it is also possible to forward alarm messages from the GUI, LE Notif GUI, to a law enforcement backend system, LE Backend.

[0120] Each cell, Cell 1510, Other Cell 1510', consists of a Host Application 1520 and one

or more Coprocessors (Coprocessor 1540, Other Coprocessor). The host application 1520, receives requests via the network handler 1523, to which the interfaces connect. The network handler 1523, forwards requests to one of the coprocessors 1540, 1540' in this cell 1510 via the host-card communication component 1524. Another component of the host application is the storage handler 1522, which receives database queries from the coprocessor 1540, via the host-card communication component 1524 and delivers them to local or remote databases 1521. Inside the coprocessor 1540, requests coming from the network handler 1523 are processed by the network channel handler 1541, which implements encryption/decryption, authentication, and sessions and forwards the clear-text request to a request handler 1543. The request handler 1543 checks the request for completeness and correctness and then delegates it to one of the specialized handlers. The screening request handler 1546, processes screening requests, the configuration request handler 1545, allows for updates of control structures inside the service, the update request handler 1544, can change the contents of the watch-lists, which are used for screening, and the inter-cell handler 1550, is used to forward requests to other cells 1510' and to distribute data to other cells. The inter-cell handler 1550 is controlled by a global cell model 1549, which gets feedback about network quality from the request handler 1543, and thus dynamically learns about the best routes through the network. Most handlers need access to configuration information, which is provided by the configuration handler 1547, and access to the database 1521 on the host, which is provided by the bucket handler 1548, which in turn uses the database channel handler 1542. The database channel handler 1542, handles all encryption/decryption and authentication of data from and to the database 1521.

[0121] Figs. 12A – 12C show different possible designs according to other embodiments of the present invention. Figures 12A through 12C show different possible designs for a single cell: single tier Figure 12A, two tier Figure 12B, two tier with replicated application servers Figure 12C. In the single tier environment a cell will include a firewall 2011 and a host application server 2010. In the two tier case, firewalls 2011 and 2021 are used and the database server 2040 is separate from the host application server 2020. The configuration shown in Figure 12C includes multiple host application

servers 2020.

[0122] Another embodiment of the present invention can also reduce the dynamic features of service specifications. Yet at an extreme position, one can completely specialize the system to one service specification or a small number of different service specifications. Technically this can be done by partial evaluation of the contract engine with the known service specifications and optimization of the resulting code. This can be done for example using the approach of (Gafter, "Method and system for translating a software implementation with data-dependent conditions to a data flow graph with conditional expressions", U.S. Pat. No. 5,650,948, issued Jul. 22, 1997) hereby incorporated herein by reference in its entirety. The result is equivalent to a static implementation, in which service specifications are not explicit data objects, and where the enforcement of the agreements between the parties is hard wired in the program code. In such an embodiment, revocation of service specifications and agreement on new service specifications requires re-generation or re-implementation of parts of the software, reinstallation, and a restart of the service.

[0123] It is to be understood that the provided illustrative examples are by no means exhaustive of the many possible uses for the invention.

[0124] From the foregoing description, one skilled in the art can easily ascertain the essential characteristics of this invention and, without departing from the spirit and scope thereof, can make various changes and modifications of the invention to adapt it to various usages and conditions.

[0125] It is to be understood that the present invention is not limited to the sole embodiment described above, but encompasses any and all embodiments within the scope of the following claims